Terms used
memory stack program counter interrupt push pop bank code register

Found **32** of **178,880**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 32 Result page: **1** 2 next

Relevance scale

1 The Postroom Computer

Hugh Osborne
December 2001 **Journal on Educational Resources in Computing (JERIC)**, Volume 1 Issue 4
Publisher: ACM Press
Full text available: pdf(242.80 KB) Additional Information: full citation, abstract, references, index terms

The Postroom Computer is a computer architecture simulator based on the *Little Man Computer* developed in 1965 by Stuart Madnick and John Donovan. It provides a family of architectures suitable for use in teaching introductory computer architectures. It is designed to introduce aspects of computer architecture and low-level programming in an incremental way. The extensions are designed to provide a range of computing models within the Little Man Computer paradigm. As they are introduced th ...

Keywords: Computer architecture simulator, education

2 Static checking of interrupt-driven software

Dennis Brylow, Niels Damgaard, Jens Palsberg
July 2001 **Proceedings of the 23rd International Conference on Software Engineering**
Publisher: IEEE Computer Society
Full text available: pdf(157.76 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Resource-constrained devices are becoming ubiquitous. Examples include cell phones, palm pilots, and digital thermostats. It can be difficult to fit required functionality into such a device without sacrificing the simplicity and clarity of the software. Increasingly complex embedded systems require extensive brute-force testing, making development and maintenance costly. This is particularly true for system components that are written in assembly language. Static checking has the potential o ...

3 Eliminating stack overflow by abstract interpretation

John Regehr, Alastair Reid, Kirk Webb
November 2005 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 4 Issue 4
Publisher: ACM Press
Full text available: pdf(510.78 KB) Additional Information: full citation, abstract, references, index terms

An important correctness criterion for software running on embedded microcontrollers is *stack safety*: a guarantee that the call stack does not overflow. Our first contribution is a method for statically guaranteeing stack safety of interrupt-driven embedded software using an approach based on context-sensitive dataflow analysis of object code. We have implemented a prototype stack analysis tool that targets software for Atmel AVR microcontrollers and tested it on embedded applications com ...

Keywords: Microcontroller, abstract interpretation, call stack, context sensitive, dataflow analysis, interrupt-driven, sensor network

4 Information structure models: A data structure model of the B6700 computer system

E. I. Organick, J. G. Cleary
February 1971 **ACM SIGPLAN Notices**, Volume 6 Issue 2
Publisher: ACM Press
Full text available: pdf(6.95 MB) Additional Information: full citation, abstract, references

The coupling of the B6700 hardware and the data structures characteristic of the computations on which it (the B6700) must operate is described from two points of view. The body of the paper gives a conceptual (Contour) model view and introduces hardware concepts of the B6700 with a minimum of detail; the supporting appendix, on the other hand, offers a thorough exposure to the hardware details, which are brought into focus via a global view of the B6700 operating system. A somewhat different te ...

5 Microprogrammable microprocessor survey

Phillip M. Adams
June 1978 **ACM SIGMICRO Newsletter**, Volume 9 Issue 2
Publisher: ACM Press
Full text available: pdf(1.24 MB) Additional Information: full citation, abstract

The Motorola M10800 LSI processor family consists of a sequencer, referred to as a Microprogram Control Function (MCF) - MC10801, and a processing element, referred to as a 4-bit ALU Slice - MC10800 (Not to be confused with the processor family number M10800). Undoubtedly, the most interesting feature of the M10800 processor family is the ECL technology used to produce it. The M10800 processor family is completely MECL 10,000 compatible and exhibits the ultra-high speed performance of ECL logic.

6 List processing in real time on a serial computer

Henry G. Baker
April 1978 **Communications of the ACM**, Volume 21 Issue 4
Publisher: ACM Press
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

A real-time list processing system is one in which the time required by the elementary list operations (e.g. CONS, CAR, CDR, RPLACA, RPLACD, EQ, and ATOM in LISP) is bounded by a (small) constant. Classical implementations of list processing systems lack this property because allocating a list cell from the heap may cause a garbage collection, which process requires time proportional to the heap size to finish. A real-time list processing system is presented which continuously reclaims garb ...

Keywords: CDR-coding, LISP, compacting, file or database management, garbage collection, list processing, real-time, reference counting, storage allocation, storage management, virtual memory

7 Register allocation for free: The C machine stack cache

David R. Ditzel, H. R. McLellan

March 1982 **ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , Proceedings of the first international symposium on Architectural support for programming languages and operating systems ASPLOS-I**, Volume 17 , 10 Issue 4 , 2

Publisher: ACM Press

Full text available: pdf(673.92 KB) Additional Information: full citation, abstract, references, citings, index terms

The Bell Labs C Machine project is investigating computer architectures to support the C programming language.1 One of the goals is to match an efficient architecture to the language and the compiler technology available. Measurements of different C programs show that roughly one out of every twenty instructions executed is either a procedure call or return.2 Procedure call overhead is therefore a very important consideration in the overall machine ...

8 The effect of the Harvard architecture on the teaching of assembly language

Ward Douglas Maurer

May 2005 **Journal of Computing Sciences in Colleges**, Volume 20 Issue 5

Publisher: Consortium for Computing Sciences in Colleges

Full text available: pdf(162.90 KB) Additional Information: full citation, abstract, references, index terms

We present here the description of a first assembly language course which uses chips with a modified Harvard architecture, rather than the usual von Neumann architecture. This means that instruction and data memory are separate, allowing instruction sizes to be unrestricted to multiples of 8. These are microcontroller chips, which are the ones in cars, toys, appliances, and the like. There are more of them today than any other computer, which motivates our interest in teaching them. Also, unlike ...

9 The microarchitecture of a capability-based computer

D. A. Abramson, J. Rosenberg

December 1986 **ACM SIGMICRO Newsletter , Proceedings of the 19th annual workshop on Microprogramming MICRO 19**, Volume 17 Issue 4

Publisher: ACM Press

Full text available: pdf(831.64 KB) Additional Information: full citation, abstract, references, index terms

This paper describes the micro-architecture of a microprogrammed workstation called MONADS-PC. The system has been specifically designed to support a very large uniform virtual memory, capability-based addressing and information hiding software modules with procedural interfaces. The paper gives a brief introduction to these topics followed by implementation details of the system.

10 Sh-BOOM: the sound of the RISC market changing

George William Shaw

March 1991 **Proceedings of the second and third annual workshops on Forth**

Publisher: ACM Press

Full text available: pdf(686.65 KB) Additional Information: full citation, index terms

11 Evaluation of a concurrent error detection method for microprogrammed control units

A. Bailas, L. L. Kinney

January 1988 **Proceedings of the 21st annual workshop on Microprogramming and microarchitecture**

Publisher: IEEE Computer Society Press

Full text available: Additional Information:

pdf(1.33 MB) full citation, references, index terms

12 Chap - a SIMD graphics processor

Adam Levinthal, Thomas Porter

January 1984 **ACM SIGGRAPH Computer Graphics , Proceedings of the 11th annual conference on Computer graphics and interactive techniques SIGGRAPH '84**, Volume 18 Issue 3

Publisher: ACM Press

Full text available: pdf(533.22 KB) Additional Information: full citation, abstract, references, citings, index terms

Special purpose processing systems designed for specific applications can provide extremely high performance at moderate cost. One such processor is presented for executing graphics and image processing algorithms as the basis of a digital film printer. Pixels in the system contain four parallel components: RGB for full color and an alpha channel for retaining transparency information. The data path of the processor contains four arithmetic elements connected through a crossbar netw ...

Keywords: Compositing, Computer graphics, Digital film printers, Parallel processing, SIMD architecture, Tesselation

13 Swamp: a fast processor for Smalltalk-80

David M. Lewis, David R. Galloway, Robert J. Francis, Brian W. Thomson

June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPLSA '86**, Volume 21 Issue 11

Publisher: ACM Press

Full text available: pdf(849.10 KB) Additional Information: full citation, abstract, references, citings, index terms

A processor for the Smalltalk-80† programming language is described. This machine is implemented using a standard bit slice ALU and sequencer, TTL MSI, and NMOS LSI RAMS. It executes an instruction set similar to the Smalltalk-80 virtual machine instruction set. The data paths of the machine are optimized for rapid Smalltalk-80 execution by the inclusion of a context cache, tag checking, and a hardware method cache. Each context is only partly initialized when created, and has no memor ...

14 μ3L: An HLL-RISC processor for parallel execution of FP-language programs

M. Castan, E. I. Organick

April 1982 **Proceedings of the 9th annual symposium on Computer Architecture**

Publisher: IEEE Computer Society Press

Full text available: pdf(709.88 KB) Additional Information: full citation, abstract, references, index terms

To eliminate the conceptual distance between the hardware instruction set and the user interface, some architects advocate High Level Language (HLL) machines. To obtain simple, fast and cheap machines, some architects advocate Reduced Instruction Set Computer (RISC) machines. This paper reconciles both views and presents an architecture which has both an HLL user interface and a RISC hardware. Each instance of this architecture is a module of an HLL multiprocessor system. Functio ...

15 A RISC architecture for symbolic computation

Richard B. Kieburtz

October 1987 **ACM SIGARCH Computer Architecture News , ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the second**

international conference on Architectual support for programming languages and operating systems ASPLOS-II, Volume 15 , 22 , 21 Issue 5 , 10 , 4
Publisher: IEEE Computer Society Press, ACM Press

Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings, index terms

The G-machine is a language-directed processor architecture designed to support graph reduction as a model of computation. It can carry out lazy evaluation of functional language programs and can evaluate programs in which logical variables are used. To support these language features, the abstract machine requires tagged memory and executes some rather complex instructions, such as to evaluate a function application.This paper explores an implementation of the G-machine as a high performance RI ...

16 The reduction of branch instruction execution overhead using structured control flow
Robert G. Wedig, Marc A. Rose
January 1984 **ACM SIGARCH Computer Architecture News , Proceedings of the 11th annual international symposium on Computer architecture ISCA '84**, Volume 12 Issue 3
Publisher: ACM Press

Full text available: pdf(707.36 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a technique for specifying change of control (e.g. branch) commands at a sequential processor's macroinstruction set level. It is shown that by representing high level language (HLL) control statements with special machine language instructions, the usual delays associated with control flow changes can be reduced. Preserving the HLL control flow information increases performance by reducing both the number of executed branches and pipeline breaks.

17 Evolving minicomputer architecture
T. G. Lewis
October 1977 **ACM SIGMINI Newsletter**, Volume 3 Issue 4
Publisher: ACM Press
Full text available: pdf(951.85 KB) Additional Information: full citation, references

18 Environments for monitoring and dynamic analysis of execution
Birol O. Aygun
June 1973 **Proceedings of the 1st symposium on Simulation of computer systems**
Publisher: IEEE Press
Full text available: pdf(1.52 MB) Additional Information: full citation, abstract, references, index terms

The research underlying this report is concerned with improving the lot of the programmer in debugging, analyzing and modelling his, or others', programs. It is my belief that the area of "execution analysis tools" has been neglected in the formative stages of system planning in contemporary systems. This has led to the design of many software and hardware "probes", "performance monitors", debugging aids, etc. which have been forced to work with uncoopera ...

19 Towards an efficient, machine-independent language for microprogramming
David A. Patterson, Karl Lew, Richard Tuck
November 1979 **ACM SIGMICRO Newsletter , Proceedings of the 12th annual workshop on Microprogramming MICRO 12**, Volume 10 Issue 4
Publisher: IEEE Press, ACM Press
Full text available: pdf(913.17 KB) Additional Information: full citation, abstract, references, citings, index terms

A machine independent low level language YALLL is presented. This language produces microcode for two very different machines: Hewlett Packard HP 300 and Digital Equipment Corporation VAX 11/780. The efficiency of this language is tested by comparing two examples on both machines to microassembly coded versions. To our best knowledge, this is the first time programs have been compiled and executed on two different microarchitectures. These examples also let us compare the efficiency of the ...

20 Burroughs Corporation: corporate public relations

October 1974 **ACM SIGMICRO Newsletter**, Volume 5 Issue 3

Publisher: ACM Press

Full text available: pdf(7.37 MB) Additional Information: full citation

Results 1 - 20 of 32 Result page: **1** 2 next

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	22717360	@ad<"20020719"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:04
L2	0	(Adamo near Carlo).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:05
L3	4	(Adamo near4 Carlo).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:05
L4	141	711/132.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:08
L5	29577	"711"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:08
L6	9	(Santi near Carlo).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L7	6	(Edmondo near Gangi).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L8	13	L6 or L7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L9	2	L6 and L7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09

EAST Search History

L10	2147	memory adj stack	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L11	2037	data adj stack	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L12	16608	program adj counter	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L13	834	condition adj code adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L14	666068	interrupt$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L15	7540	(push$3 or pop$3) adj operation	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L16	19515	memory near2 bank$2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L17	666068	interrupt$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L18	7540	(push$3 or pop$3) adj operation	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L19	1022	L17 and L18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09

EAST Search History

L20	834	condition adj code adj register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L21	1022	L17 and L18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L22	20	L21 and L20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L23	16608	program adj counter	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L24	20	L21 and L20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L25	11	L23 and L24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L26	2147	memory adj stack	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L27	11	L23 and L24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09
L28	2	L26 and L27	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/25 14:09

IEEEXplore# Search History

IEEE Xplore RELEASE 2.1

Welcome United States Patent and Trademark Office

BROWSE SEARCH IEEE XPLORE GUIDE

Search Session History

Sun, 25 Jun 2006, 2:53:32 PM EST

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Search Query Display

Recent Search Queries

#1	((memory stack)<in>metadata)
#2	(CCR or (conditional code register)<IN>metadata)
#3	(CCR or (conditional code register)<IN>metadata)
#4	(program counter<IN>metadata)
#5	(interrupt* routine<IN>metadata)
#6	(interrupt* routine<IN>metadata)
#7	((push or pop) and operation<IN>metadata)
#8	(memory bank<IN>metadata)
#9	(data stack<IN>metadata)
#10	(((memory stack)<in>metadata)) <AND> ((CCR or (conditional code register) <IN>metadata))
#11	((((memory stack)<in>metadata)) <AND> ((CCR or (conditional code register) <IN>metadata)))
#12	(((((memory stack)<in>metadata)) <AND> ((CCR or (conditional code register) <IN>metadata)))) <AND> ((interrupt* routine<IN>metadata))
#13	(((((memory stack)<in>metadata)) <AND> ((CCR or (conditional code register) <IN>metadata)))) <AND> ((memory bank<IN>metadata))
#14	((((((memory stack)<in>metadata)) <AND> ((CCR or (conditional code register) <IN>metadata)))) <AND> ((memory bank<IN>metadata))) <AND> ((data stack<IN>metadata))